A Novel Approach to Evaluate the Top K Interesting Ranks from Web Search Engines

Chien-I Lee, Cheng-Jung Tsai, Yu-Chiang Li and Cheng-Tao Wu

Institute of Information Education
National Tainan Teachers College, Tainan, Taiwan, R.O.C.
Email:leeci@ipx.ntntc.edu.tw

Abstract

In recent years, several search engines had been developed to help people find interesting information among the rapidly increasing number of web pages in the Internet. To obtain useful and reasonable searching results, users may submit queries with more than one query terms combined by a Boolean expression, which has been supported by all existing search engines. However, these search engines all put the same emphases on every query term combined by the Boolean expression. In other words, all search engines nowadays do not consider that users may want to put more emphases on one term rather than on the others. In this paper we propose a novel approach, called Extreme Score Analysis method (ESA method), to solve this problem. Our ESA method can efficiently find the top K ($K > 0$) interesting web pages when the user assigns different weights for query terms.

1. Introduction

As the fast process of computer and network technologies, computers connected in the Internet will soon become the indispensable electrical appliances of our daily life. Through the Internet, people can get their desired information much easier and quicker. But, meanwhile, the rapidly growing data carried on the Internet really make people very difficult to search and filter the appropriate information they want. In recent years, several *search engines* [3, 14, 15, 17, 22, 24] had been developed to reduce such overloads.

In general, in order to support a full-text information retrieval, the search engines filter terms in the contents of web pages to establish a corresponding *inverted index* [23]. When users submit a query with some proper query terms as keywords (these query terms can be combined by a *Boolean expression*), the search engines will look the terms up in the inverted index to find out the corresponding web pages, and further rank these web pages by their own predetermined *term weighting functions* and *vector similarity functions*. Finally, users can view those ranked web pages and get more relevant information in the front of them.

Usually, users may submit queries with more than one query terms combined by a Boolean expression to enlarge or narrow their searching interests. Although the search engines nowadays all support the Boolean expression, they put the same emphases on all the query terms combined by the Boolean expression. For example, when a user submits a Boolean expression with two query terms, "music and download", a typical search engine will return the web pages which contain both terms "music" and "download", and ranks these web pages according to their scores, which could be the sum of the individual *contributed values* for the two query terms in their own context. (Note that the contributed value for a term denotes the value returned from its term weight function.) However, users may put more emphases on one term rather than on the other. Continuing with the foregoing example, users may want to look for the web pages, which contain "music" for "download", but put more emphases on "music" rather than on "download". For quantify such a case, users can be allowed to assign different weights to each query term in the Boolean expression, e.g. "(0.8) music and (0.2) download", which means the degrees of importance of the two query terms are in a ratio of 4:1.

To fulfill such a requirement for different weights of query terms, it is necessary to re-calculate the final score for each web page according to the new given weights. After this re-calculation of web pages' scores, users can get the true ranking results. However, a typical search engine merely returns matched web pages with final scores, ranks, etc., but without the individual contributed value for each query term. Therefore, to re-calculate the final scores, there are two problems needed to be resolved. First, we need to know the scoring function of the search engine. Usually, due to commercial secret, we usually have no idea of the search engine's scoring function. Basically, we can solve this problem by adopting another scoring function, which can be defined by ourselves. Even so, there still exists the second problem is that we have to scan every returned web page's content to get each query term's contributed value. Such a task will be extremely time-consuming. Moreover, in a real situation, for the sake of time or no patience, most of users usually only view the top K web pages from those HN returned web pages [4] ($K << HN$, where HN is the total number of returned web pages).

To avoid the overhead of re-calculating the whole HN returned web pages, in this paper we will propose a novel approach, called *Extreme Score Analysis Method* (*ESA method*), which can find the top K interesting web pages without re-scanning the HN returned web pages to get the individual query terms' contributed values. Instead, our method will inform users that the top K interesting web pages they request will be among the top R returned web

pages ($K \le R << HN$). From the performance study, we can find that the value of R will be close to that one of K. For example, if one user submits a query with two terms and wants to get the top K web pages according to his/her weights (0.6 and 0.4, respectively) of query terms, the value of R will be approximately equal to $1.5 \times K$.

The rest of the paper is organized as following. In Section 2, we survey some basic components of a typical search engine and some related term weighting functions and vector similarity functions. Section 3 describes the basic idea of our Extreme Score Analysis Method. Section 4 presents an experimental evaluation of our method. Finally, Section 5 is the conclusion.

2. The Components of Search Engines

In general, a typical search engine is composed of three components:

(1) Indexer Robot

The indexer robot [5] is an autonomous WWW browser, which communicates with WWW servers using HTTP (Hypertext Transfer Protocol). It visits a given WWW site, traverses hyperlinks in a breadth-first manner, retrieves WWW pages, extracts keywords and hyperlink data from the pages, and inserts the terms and hyperlink data into an index. A list of target sites is given to the indexer robot for creating the index initially.

(2) Indexer database

The Indexer robot reads web pages and sends these web pages to the indexer database to crate indexing records. This indexer database is also named an inverted index. Each web page in the inverted index is represented as a vector $d = (d_1,\ldots,d_m)$, where d_i is the weight of the ith term t_i in representing the web page, $1 \le i \le m$. If the original web page is changed, the inverted index should be updated. The update is not made in the inverted index until the indexer robot has revisited the changed web page again.

(3) Ranking

In order to rank all related web pages, the search engine assign relevance scores to them [18]. The scores indicate the similarities between some given query terms, which can be similarly represented as a vector $q = (q_1,\ldots,q_m)$ (where q_i is the weight of the ith term t_i , $1 \le i \le m$), and the web pages. However, each search engine may employs a characteristic *term weighting function*, which assign weight to each term, and a characteristic *vector similarity function*, which assign relevance scores to web pages.

A well-used term weighting function is called *tf×idf* [11], which assumes that term importance is proportional to the standard occurrence frequency of each term k in each web page H_i (that is, $FREQ_{ik}$) and inversely proportional to the total number of web pages in the web page collection (that is, $HOPFREQ_i$) to which each term is assigned. Then, a general term weighting function of tf×idf can be

$$WEIGHT_{ik} = FREQ_{ik} \times [\log_2(n) - \log_2(HOPFREQ_k) + 1],$$

where n is the total number of web pages in the collection.

There were many others proposed term weighting functions, such as *Signal-Noise Ratio* [6] and *Term Discrimination Value* [21]. Although every term weighting function has its own properties, all of them propose the same hypothesis that term importance is proportional to the standard occurrence frequency of each term in each web page as the one in tf×idf. Nevertheless, tf×idf is superior to the others in the aspects of efficiency and cost [12, 18]. Furthermore, because the web pages are written in hypertext markup language (HTML), each search engine may take some related factors (e.g. hyperlink, number of times the keywords occur in the document title…etc.) into account to provide more suitable ranking results [1]. Consequently, the most term weighting function applied by the existing search engines are derived from tf×idf.

Although, there were also many kinds of vector similarity functions, they all exhibited one common property, namely that the similarity value increases when the weight of the common properties in two vectors increases. *Jaccard* and *cosine* coefficient, which are two kinds of vector similarity functions, have been widely used for the evaluation of retrieval functions [19, 20]. Both values of the two similarity functions increase when the *dot product* of two vectors increases.

3. Extreme Score Analysis method

As discussed in Section 2, we can assume that when a user submits a query with n query terms, the score KS_i (here, we call it *original score*) search engines assign to each web page H_i (i denotes the original rank of this web page among the returned matched web pages) will be

$$KS_i = \sum_{j=1}^{n} W_j(b((FREQ_{ij} + a)/HOCFREQ_j) + c),$$

where W_j is the weight of term j given in the user's query and $(b((FREQ_{ij} + a) / HOCFREQ_j) + c)$ is the contributed value of term j in KS_i. Variables a, b and c are used by each search engine to improve the tf×idf under the consideration of some possible factors. Nevertheless, what our research concern is only about the ranking result of returned web pages. Whatever the values of Variables *a, b* and *c* will be, the ranking result will never change. Furthermore, as mentioned in Section 1, all current search engines put the same emphases on every query term (that is, $W_j = 1 / n$) Therefore, the original score can be simplified as

$$KS_i = \sum_{1}^{n} (1/n) \times ConV_{ij},$$

where $ConV_{ij} = b((FREQ_{ij} + a) / HOCFREQ_j) + c$.

However, users may assign each term weight unequally. Obviously, alterations in term weights will cause the changes of original scores. The score that is re-computed from the original score is called *target scores* KT_i and can be defined as

$$KT_i = \sum_{1}^{n} W_j \times ConV_{ij},$$
$$\text{where } \exists\ j, W_j \neq 1/n.$$

For example, suppose a user submits a query with two query terms, term p and term q, combined by a Boolean expression and the user want to assign 0.7 and 0.3 to each term's weight, respectively. Then, the original score of the ith matched web page will be

$$KS_i = 0.5ConV_{pi} + 0.5ConV_{qi}.$$

And, the target score of the ith matched web page will be

$$KT_i = 0.7ConV_{pi} + 0.3ConV_{qi}.$$

To be convenient to explain our method, we will first consider the case of two query terms, term p and term q. Without loss of generality, we divide $ConV_p$ and $ConV_q$ by their maximum respectively to limit their values between 0 and 1, i.e., $0 \leq ConV_p, ConV_q \leq 1$. Then, there are some theorems, which are applied in our method, need to be stated and proved in the following.

Theorem 1. *Let x-axis and y-axis denote $ConV_p$ and $ConV_q$ respectively in the coordinate plane. Then a linear equation, which passes through the origin and is orthogonal to the straight line $SC = W_p \times ConV_p + W_q \times ConV_q$, will be $ConV_q = (W_q / W_p) \times ConV_p$, where $0 \leq W_p, W_q \leq 1$.*

Proof. According to *Slope Theorem*, the product of slopes of two straight lines will be –1 if they are orthogonal to each other. Therefore, we can obtain the slope of the straight line, which is orthogonal to the straight line SC, is W_q / W_p. Moreover, because the straight line passes through the origin, we can obtain the linear equation will be $ConV_q = (W_q / W_p) \times ConV_p$. □

Theorem 2. *Let $SC_1 = W_p \times ConV_p + W_q \times ConV_q$ and $SC_2 = W_p \times ConV_p + W_q \times ConV_q$ represent two parallel lines in the Coordinate Plane and intersect the straight line $ConV_q = (W_q / W_p) \times ConV_p$ in the point A and point B, respectively. If the length of line segment $\overline{OA}$ > the length of line segment $\overline{OB}$, then $SC_1 > SC_2$.*

Proof. First, we can obtain the coordinates of A is $((SC_1 \times W_p) / (W_p^2 + W_q^2), (SC_1 \times W_q) / (W_p^2 + W_q^2))$, and that of B is $((SC_2 \times W_p) / (W_p^2 + W_q^2), (SC_2 \times W_q) / (W_p^2 + W_q^2))$. Then, we can obtain the length of line segment $\overline{OA} = SC_1 / (W_p^2 + W_q^2)^{1/2}$ and $\overline{OB} = SC_2 / (W_p^2 + W_q^2)^{1/2}$. Clearly, because of $\overline{OA} > \overline{OB}$, we can infer that $SC_1 > SC_2$. □

From Theorems 1 and 2, we can obtain Figure 1 and Figure 2 as shown in the following. In both figures, x-axis and y-axis represent the contributed values of query terms p and q, respectively. The point $(ConV_{pi}, ConV_{qi})$ in the coordinate plane denotes the web page H_i returned by the search engine, and every dashed straight line which intersects the point $(ConV_{pi}, ConV_{qi})$ denotes the score of each web page H_i when the term weights are W_p, W_q respectively. In particular, Figure 1 represents the distribution of original scores of returned web pages, and Figure 2 represents the distribution of target scores. From the comparison of Figure 1 and Figure 2, we can observe that the dashed line, which denotes the score of web page H_i, will alter when the weights of the two query terms vary. From Theorem 1 we can know the reason is that original score considers the weights of both term are equivalent (that is, $W_q / W_p = 1$), but target score does not (that is, $W_q / W_p \neq 1$). As well, from Theorem 2 we can obtain that the further the distance between the dashed straight line and the origin is, the higher the score denoted by the dashed straight line will be.

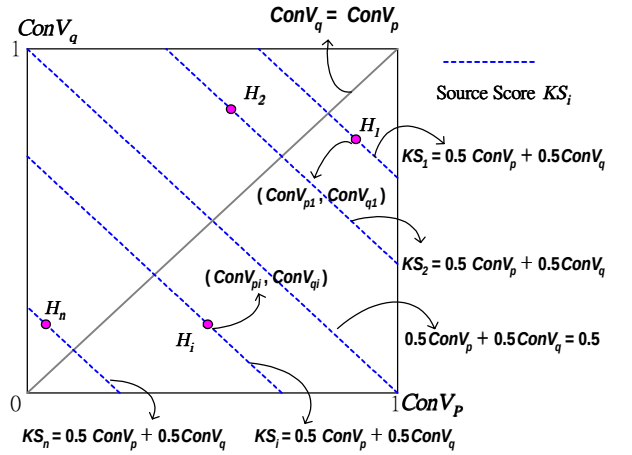

Figure 1. The distribution of original scores in the coordinate plane.

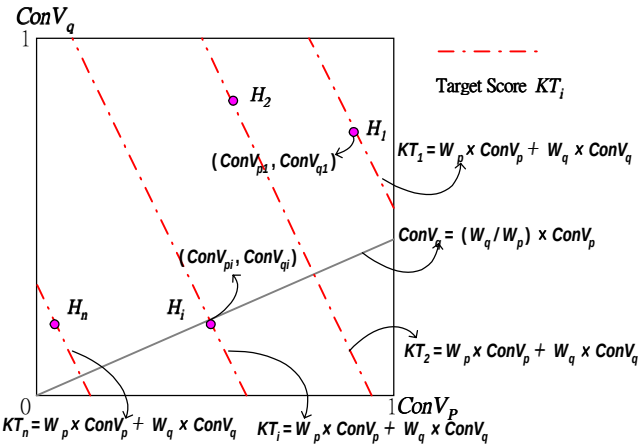

Figure 2. The distribution of target scores in the coordinate plane.

Theorem 3. *As shown in Figure 3, let $KS_i = 0.5ConV_p + 0.5ConV_q$ denotes a straight line in the coordinate plane, where $0 \leq ConV_p, ConV_q \leq 1$ and $0 < W_q < W_p < 1$. Then, we can obtain an infinite number of straight lines $KT_i = W_p \times ConV_p + W_q \times ConV_q$ which intersect KS_i and are orthogonal to the straight line $ConV_q = (W_q / W_p) \times ConV_p$. Furthermore, we can obtain KT_{maxi} and KT_{mini}, which satisfy $KT_{maxi} \leq KT_i \leq KT_{mini}$, for each KS_i, respectively.*

Proof. Solve simultaneously the set of Equations (1) and (2):

$$KS_i = 0.5ConV_p + 0.5ConV_q, \quad (1)$$
$$KT_i = W_p \times ConV_p + W_q \times ConV_q. \quad (2)$$

By (1) × $2W_p$ - (2), we can obtain

$$ConV_q \times (W_p - W_q) = 2W_p \times KS_i - KT_i.$$
$$ConV_q = (2W_p \times KS_i - KT_i) / (W_p - W_q) \quad (3)$$

By (1) × $2W_q$ - (2), we can obtain

$$ConV_p \times (W_q - W_p) = 2W_q \times KS_i - KT_i.$$
$$ConV_p = (2W_q \times KS_i - KT_i) / (W_q - W_p). \quad (4)$$

$$0 \le ConV_p, ConV_q \le 1. \quad (5)$$
$$1 < W_p < W_q < 0. \quad (6)$$

By solving simultaneously Equations (3), (5), and (6), we can obtain

$$0 \le (2W_p \times KS_i - KT_i) / (W_p - W_q) \le 1.$$
$$0 \le (2W_p \times KS_i - KT_i) \le W_p - W_q.$$
$$W_q + W_p(2KS_i - 1) \le KT_i \le 2W_p \times KS_i.$$
$$W_q + W_p(2KS_i - 1) \le KT_i \le W_p + W_p(2KS_i - 1). \quad (7)$$

Similarly, by solving simultaneously Equations (4), (5), (6),we can obtain

$$W_q + W_q(2KS_i - 1) \le KT_i \le W_p + W_q(2KS_i - 1). \quad (8)$$

Finally, by solving simultaneously Equations (7) and (8), we can obtain the maximum of KT_i (that is, $KT_{maxi} = Min(W_p + W_q(2KS_i - 1), W_p + W_p(2KS_i - 1))$ and the minimum of KT_i (that is, $KT_{mini} = Max(W_q + W_q(2KS_i - 1), W_q + W_p(2KS_i - 1))$, which satisfy $KT_{maxi} \le KT_i \le KT_{mini}$ for each KS_i. □

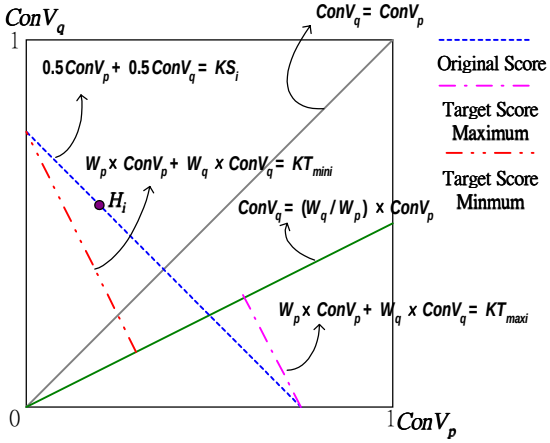

Figure 3. The relation between original scores and target scores.

Theorem 4. *Continuing with Theorem 3, let* $KS_i = 0.5ConV_p + 0.5ConV_q$ *and* $KS_j = 0.5ConV_p + 0.5ConV_q$ *represent two straight lines in the coordinate plane. If* $KS_i > KS_j$, *then* $KT_{mini} > KT_{minj}$ *and* $KT_{maxi} > KT_{maxj}$.

Proof. From Equations (7) and (8) in Theorem 3, we can get that if the values of term weights W_p and W_q are fixed, then the larger KS_i is, the larger KT_{maxi} and KT_{mini} will be. Therefore if $KS_i > KS_j$, then $KT_{mini} > KT_{minj}$ and $KT_{maxi} > KT_{maxj}$. □

Theorem 5. *For every i, suppose there are* $KT_{maxi} \ge KT_i \ge KT_{min}$ $(1 \le i \le HN)$, *and satisfy* $KT_{max1} \ge KT_{max2} \ge \ldots \ge KT_{maxi} \ge \ldots \ge KT_{maxHN}$, $KT_{min1} \ge KT_{min2} \ge \ldots \ge KT_{mini} \ge \ldots \ge KT_{minHN}$. *If there is a* KT_{maxj} *that satisfies* $KT_{maxj} \ge KT_{mink}$, *then the* KT_j *may be one of the top K of all* KT_i.

Proof. Suppose there is a KT_j $(j > K)$ and satisfies $KT_{maxj} \ge KT_{minK}$, but, KT_j should not be one of the top K of all KT_i. However, as asserted above, Because of $KT_{min1} \ge KT_{min2} \ge \ldots \ge KT_{minK-1} \ge KT_{minK}$, the top K of all KT_i will be KT_1, KT_2, KT_3,…, KT_K in turn when $KT_i = KT_{mini}$, $1 \le i \le K$. Nevertheless, because of $KT_{maxj} \ge KT_{minK}$, KT_j may be bigger than KT_K and becomes the Kth rank of *all* KT_i when $KT_j = KT_{maxj}$. As a result, our previous assumption is not true. Consequently, for each i if $KT_{maxi} \ge KT_{mink}$, then the KT_i may be one of the top K of all KT_i. □

As stated before, since the search engine do not return the individual contributed values $ConV_{pi}$ and $ConV_{qi}$ for each returned web page H_i, the coordinates of web page H_i, $(ConV_{pi}, ConV_{qi})$, may locate at anywhere on the straight line $KS_i = 0.5ConV_{pi} + 0.5ConV_{qi}$. As a result, we can not obtain the actual target score KT_i for H_i.

From Theorem 3, we know that KT_i will be between KT_{mini} and KT_{maxi}, as shown in Figure 3. That is, for given term weight values W_p and W_q, which are assigned by users, we can derive a value interval of target score from each original score. Each value interval will have a pair of *maximum target score* KT_{maxi} and *minimum target score* KT_{mini}. Furthermore, from Theorem 4 we can observe that the distribution of KT_{maxi} and KT_{mini} decrease gradually with the decrease in original score. Figure 4 shows such a case. Finally, from Theorem 5, to support the correct top K interesting web pages from the returned web pages, all the web pages whose $KT_{maxi} \ge KT_{minK}$ might one of the top K target ranks of web pages. That is, the user has only to view those web pages to get the real top K target ranks, instead of viewing the whole HN returned web pages.

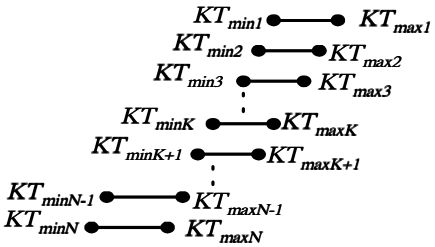

Figure 4. The distribution of KT_{maxi} and KT_{mini}.

In general, suppose a user submit a query with n different query term weights through a search engine, which returned HN matched web pages, and the user wants the top K interesting ranks of web pages according to the submitted term weights. Without loss of generality, we let

$W_1 \geq W_2 \geq, \ldots \geq W_{n-1} \geq W_n$, and the analysis steps to evaluate the required top R web pages ($R \geq K$) needed to be viewed by the user to get the real top R ranks in our proposed ESA method are shown in Figure 5. In Figure 5, to simply the expressions, we define two functions:

$$S_1(j, n) = \begin{cases} 0 & \text{if } n < j + 1, \\ \sum_{m=j+1}^{n} W_m & \text{if } n \geq j + 1; \end{cases}$$

$$S_2(1, j) = \begin{cases} 0 & \text{if } j < 2, \\ \sum_{m=1}^{j-1} W_m & \text{if } j \geq 2. \end{cases}$$

Step1:

/* In this step, ESA method will compute the KT_{maxi} and KT_{mini} for *each* returned web page i.*/

For i = 1 to HN
  For j = 1 to n
    $MinKT[j] = W_j \times n \times KS_i - (n - j)\ W_j + S_1(j, n)$;
    $MaxKT[j] = W_j \times n \times KS_i - (j - 1)\ W_j + S_2(1, j)$;
  End for
  $KT_{mini} = Max(MinKT[j])$;
  $KT_{maxi} = Min(MaxKT[j])$;
End for

Step2 :

/* The top K interesting web pages will exist in the front of the top R ranks of returned HN web pages. */

For i = 1 to HN
  If $KT_{maxi} < KT_{minK}$ Then
    $R = i - 1$;
    Go to Step3;
  End if
End for

Step3:

If the user would like to get the real top K target ranks then
  For i = 1 to R
    Retrieve H_i and compute the contributed value for each query term j;
    Score the web page H_i according to the new computed contributed value;
  End for
  Rank H_i according to the target scores and return the top K web pages to the user;
Else
  Return the top R original ranks of returned web pages to the user;
End if

Figure 5. Extreme Score Analysis method (ESA method).

There is an example as shown in Table 1 to explain the steps in our ESA method. For simplicity, we consider the case with n = 2 and the search engine returned 20 related web pages. Suppose the user wants to assign 0.3 and 0.7 to each term weight, respectively, and requests the top 3 interesting web pages. That is, n = 2, HN = 20, W_1 = 0.3, W_2 = 0.7 and K = 3. To understand and verify the result of our ESA method, we used a generator to generate $ConV_q$ and $ConV_p$ randomly for these 20 web pages and then compute each web page's original score KS_i. Furthermore, the actual value of each KT_i can be obtained as shown in Table 1. (Note that in a real situation, $ConV_q$, $ConV_p$, and KT_i are unavailable.) In Step 1, ESA method computes the KT_{maxi} and KT_{mini} for every returned web page. Then, in Step 2, ESA method will derive the total number R of web pages that the user has to view. In other words, it tries to find the web pages whose $KT_{maxi} \geq KT_{minK}$ (= 0.854). In this example, we can find that KT_{max7} (= 0.866) > KT_{minK}, but KT_{max8} does not. Finally, in Step 3, the user has only to view the top R = 7 returned web pages (i.e., H_1, H_2, H_3, H_4, H_5, H_6, H_7) to get the real top 3 interesting web pages with the new given term weights. However, the user also can wait the system to re-scan only the top R (= 7) original ranks of web pages to provide the real top 3 interesting web pages (i.e., H_1, H_3, H_5), without re-scanning all the HN = 20 returned web pages.

Table 1. An example of ESA method (n = 2, HN = 20, W_1 = 0.3, W_2 = 0.7 and K = 3)

Serial numbers	Rank of KS_i	KS_i	KT_{mini}	KT_{maxi}	$ConV_p$	$ConV_q$	KT_i
H_1	1	0.934	0.908	0.96	0.92	0.948	0.928
H_2	2	0.906	0.868	0.944	0.87	0.942	0.892
H_3	3	0.896	0.854	0.938	0.98	0.812	0.93
H_4	4	0.86	0.804	0.916	0.78	0.94	0.828
H_5	5	0.857	0.8	0.914	0.95	0.764	0.894
H_6	6	0.811	0.735	0.887	0.77	0.852	0.795
H_7	7	0.776	0.686	0.866	0.76	0.792	0.77
H_8	8	0.726	0.616	0.836	0.99	0.462	0.832
H_9	9	0.648	0.507	0.789	0.713	0.583	0.674
H_{10}	10	0.622	0.471	0.773	0.406	0.838	0.536
H_{11}	11	0.579	0.411	0.747	0.666	0.492	0.614
H_{12}	12	0.542	0.359	0.725	0.997	0.087	0.724
H_{13}	13	0.532	0.345	0.719	0.746	0.318	0.618
H_{14}	14	0.519	0.327	0.711	0.538	0.5	0.527
H_{15}	15	0.429	0.257	0.601	0.2	0.658	0.337
H_{16}	16	0.349	0.209	0.489	0.09	0.608	0.245
H_{17}	17	0.279	0.167	0.391	0.34	0.218	0.303
H_{18}	18	0.114	0.068	0.16	0.22	0.008	0.156
H_{19}	19	0.06	0.036	0.084	0.091	0.029	0.072
H_{20}	20	0.056	0.034	0.078	0.01	0.102	0.038

4. Performance evaluation

4.1 Experiment model

In Section 3, we have stated the basic idea of ESA method. In this section, because of the distribution of original score is not fixed, we establish a simulation model, which uses a generator to randomly generate the original scores for returned web pages, to evaluate the efficiencies of ESA method. There are two performance measures in our evaluation. One is *Per*, and the other is *Mul*, where *Per* denotes the percentage of web pages that need to be viewed (that is, $Per = 100 \times (R / HN)$ %), and *Mul* denotes the proportion of users' top K interesting web pages to the number of returned web pages they have to view (that is, $Mul = R / K$). Each *Per*, *R*, and *Mul* is obtained by averaging the results of simulating 100 times.

4.2 Result

The results of experiment simulation are presented in Table 2. The average of all *Mul* is 4.41, which means that on average users have only to view about 4.41 times the number of K to get their real top K interesting web pages.

When users allow the system to re-calculate the real top K target ranks, the small value of *Per* in most cases has shown the efficiency of our method again. The value of *Per* will become much larger only when the value of K is close to that of *HN*. However, search engines usually returned much more web pages than user's interesting top K web pages. That is, *HN* is usually much larger than K.

In the worst case (W_p = 0.9, W_p = 0.1, HN = 5000, and K =10), the number of web pages users have to view is about 10.8 (=*Mul*) times that of their top K interesting web pages. However, in this case, the total number of returned web pages is 5000, and the system has only to re-calculate 108 returned web pages to provide the real top K target ranks, which is still much more efficient than re-calculating all 5000 returned web pages.

Finally, we can find that in spite of what each term weight will be, the larger the *HN* is, the less the *Per* will be. And the less the difference between each query term is, the less the *Per* will be. Furthermore, the smaller the K is, the smaller the *Per* will be.

Table 2. The result of experiment simulation

W	HN	K	R_r	Per_r	Mul_r
Wp = 0.6 Wq = 0.4	5000	10	17	0.33%	1.7
		20	33	0.66%	1.65
		30	47	0.924%	1.57
	1000	10	17	1.62%	1.7
		20	32	3.10%	1.6
		30	46	4.57%	1.53
	500	10	16	3.01%	1.6
		20	31	6.07%	1.55
		30	46	9.01%	1.53
		30	45	44.3%	1.5
Wp = 0.7 Wq = 0.3	5000	10	26	0.51%	2.6
		20	50	1.00%	2.5
		30	74	1.47%	2.47
	1000	10	25	2.45%	2.5
		20	47	4.68%	2.35
		30	72	7.19%	2.4
	500	10	25	4.85%	2.5
		20	48	9.49%	2.4
		30	71	14.14%	2.37
Wp = 0.8 Wq = 0.2	5000	10	48	0.95%	4.8
		20	93	1.86%	4.65
		30	122	2.43%	4.01
	1000	10	41	4.08%	4.1
		20	81	8.01%	4.05
		30	125	12.50%	4.17
	500	10	42	8.37%	4.2
		20	78	15.54%	3.9
		30	122	24.37%	4.01
Wp = 0.9 Wq = 0.1	5000	10	108	2.16%	10.8
		20	201	4.01%	10.1
		30	280	5.59%	9.33
	1000	10	93	9.21%	9.3
		20	179	17.88%	8.95
		30	279	27.84%	9.3
	500	10	96	19.02%	9.6
		20	177	35.22%	8.85
		30	245	48.91%	8.17
Average				8.97%	4.41

5. Conclusion

The existing search engines all put the same emphases on submitted query terms combined by the Boolean expression. However, users may put different emphasis on each query term. That is, users should be allowed to assign different weight to each query term for their own search purpose. For such a case, the system must re-calculate the new score for each returned web page according to the new given weights. Because typical search engines did not return sufficient information for a system to re-calculate the new score, the re-calculating task will be time-consuming. Moreover, in a real situation, most of users usually only view the top K web pages from those HN returned web pages. Therefore, in this paper, we have proposed the ESA method to solve this problem. By our ESA method, the system does not need to re-scan the whole returned pages and can easily inform users that the top K interesting web pages they request will be among the top R web pages ($K \leq R \leq HN$). Moreover, if users want to get the actual ranking of the top K interesting web pages, the system has only to re-calculate the top R returned web pages, instead of the whole HN returned web pages. The evaluated results in Section 4 had proved the efficiency of our ESA method in the simulated environments. In the future research directions, we will extend our technique to *metasearch* [7, 8, 9, 10, 13, 16]. And, we will consider the personal factor [2] to improve the efficiency of our method.

6. References

[1] L. Allison, D.L. Dowe, G. Pringle, "What is a Tall Poppy Among Web Pages?" The Seventh International WWW Conference, Brisbane, Australia, April14-18, 1998.

[2] W.P. Birmingham, E.J. Glover, S. Lawrence, C.G. Lee, "Architecture of a Metasearch Engine that Supports User Information Needs," Proceedings of the eighth international conference on Information knowledge management, pp. 210 – 216, 1999.

[3] S. Brin, L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," Seventh International Web Conference (WWW 98). Brisbane, Australia, April 14-18, 1998.

[4] S. Chaudhuri, L. Gravano, "Evaluating Top-K Selection Queries," VLDB'99, pp. 397- 410, 1999.

[5] J. Cho, H.G. Molina, L. Page, "Efficient Crawling Through URL Ordering," Seventh International Web Conference (WWW 98). Brisbane, Australia, April 14-18, 1998.

[6] S.F. Dennis, "The Design and Testing of a Fully Automatic Indexing-Searching System for Documents Consisting of Expository Text," in Information Retrieval: A Critical Review, G. Schecter, editor, Thompson Book Co., Washington, D.C., pp.67-94, 1967.

[7] D. Dreilinger, A.E. Howe, "Experiences with Selecting Search Engines Using Metasearch," ACM Transaction on Information Systems, 15(3), pp. 195-222, 1997.

[8] O. Etzioni, E. Selberg, "MultiService Search and Comparison Using the MetaCrawer," The Fourth International Web Conference (WWW 95). Boston, USA, December 11-14, 1995.

[9] S. Gauch, M. Gomez, G. Wang, "Profusion: Intelligent Fusion from Multiple, Distributed Search Engines," Journal of Universal Computing, SpringeVelag, 2 (9), pp. 637-649, 1997.

[10] L. Gravano, Y. Papakonstantinou, "Mediating and Metasearching on the Internet," IEEE Bulletin of Data Engineering, 21 (2), pp. 28-36, 1998.

[11] K.S. Jones, "A statistical Interpretation of Term Specificity and Its Application in Retrieval," Journal of Documentation, Vol. 28, No. 1, March, pp. 11-20, 1972.

[12] F.W. Lancaster, A.J. Warner, "Information Retrieval Today, " Arlington: Information Resources Press, 1993.

[13] S. Lawrence, C.G. Lee, "Accessibility of Information on the Web," Nature, 4OO(July 8), pp. 107-109,1999.

[14] D.L. Lee, B. Yuwono, "A World Wide Web Resource Database System," IEEE Transaction on Knowledge and Data Engineering, 8 (4) , pp. 548-554, 1996.

[15] D.L. Lee, B. Yuwono, "Search and Ranking Algorithm for Locating Resources on the World Wide Web," in: Proc.12th int'l Conf. Data Engineering, pp. 164-171, 1996.

[16] K.L. Liu, W. Meng, N. Rishe, W. Wu, C. Yu, "Estimating the Usefulness of Search Engines," Proceedings of the 15th International Conference on Data Engineering, pp. 146 –153, 1999.

[17] M. Marchiori, "The Quest for Correct Information on the Web: Hyper Search Engines," The Sixth International WWW Conference (WWW 97). Santa Clara, USA, April 7-11, 1997.

[18] M.L. Mauldin, "Lycos: Design Choices in An Internet Search Service," IEEE Expert, (January-February): 8-11,1997.

[19] M. McGill, G. Salton, "Introduction to Modern Information Retrieval," New York: McGraw -Hill Book Company, 1983.

[20] W. Meng, C. Yu, "Principles of Database Query Processing for Advanced Applications," Morgan Kaufmann, San Francisco, 1998.

[21] G. Salton and C.S. Yang, "On the Specification of Term Values in Automatic Indexing," Journal of documentation, Vol. 29, No 4, December, pp.351-372, 1973.

[22] C. Schwartz, "Web search engines." Journal of the American Society for Information Science 49 (12), pp. 973-982.

[23] M.E. Senko, "File Organization and Management Information Systems," Chapter 4, Annual Review of Information Science and Technology, C. Cuadra, editor, Vol. 4, Encyclopaedia Britannica, Chicago, Illinois, pp. 111-143, 1969.

[24] M.A. Sheldon, B. Vélez, R. Weiss, "HyPursuit: A Hierarchical Network Search Engine that Exploits Content-link Hypertext Clustering," Proceedings of the Seventh ACM Conference on Hypertext, pp. 180 – 193, 1996.